

**TRIMESTER 1 2019/2020**

**TOP2121**

**OBJECT-ORIENTED PROGRAMMING**

**Objects Recognition System**

Tutorial Section: OP2B

Eugene Low Chun Chuan 1161103561

David Ang Huang Qiang 1151105412

Mosha, Joel Godfrey 1171300063

Justin John Rajoo 1161103503

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY MALAYSIA

25 SEPTEMBER 2019

**Table of Contents**

[**Introduction** 2](#_Toc20220006)

[**The UML Diagrams** 3](#_Toc20220007)

[i) Use Case Diagram 3](#_Toc20220008)

[ii) Class Diagram 4](#_Toc20220009)

[iii) State Diagram 5](#_Toc20220010)

[**Detailed Descriptions** 6](#_Toc20220011)

[**Appendix** 10](#_Toc20220012)

[i) Source Code 10](#_Toc20220013)

[ii) Sample Interface Screen 17](#_Toc20220014)

[iii) Sample Input/Output Screens 18](#_Toc20220015)

# **Introduction**

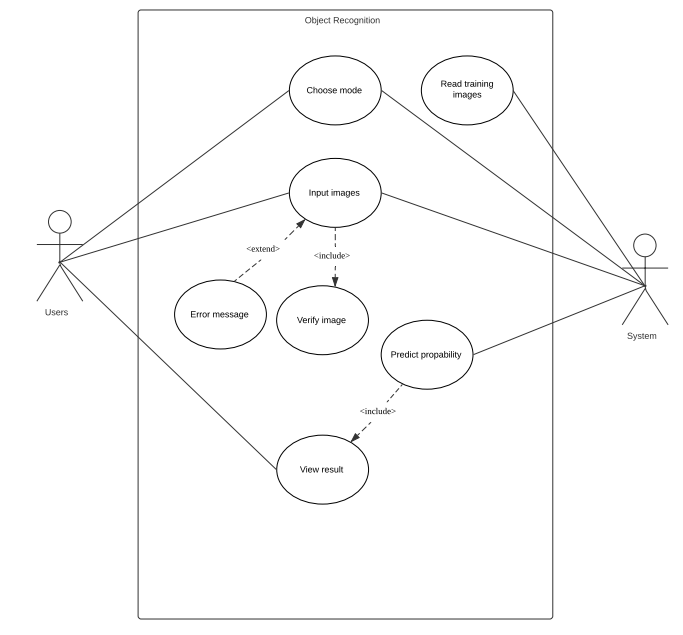
Objects recognition system finds objects in the real world from an image. This task is extremely cumbersome. Human beings can effortlessly and instantly perform object recognition. The algorithms required and implemented for the description of this task on machines is extremely difficult.

Our objects recognition system consists of pre -training module that features graph and txt file that acts as the image repository file being used to detect the objects and also training module that features a probability value calculation that is mapped by increasing its dimension values and the flatten dimension that flattens images from 2D to 1D.

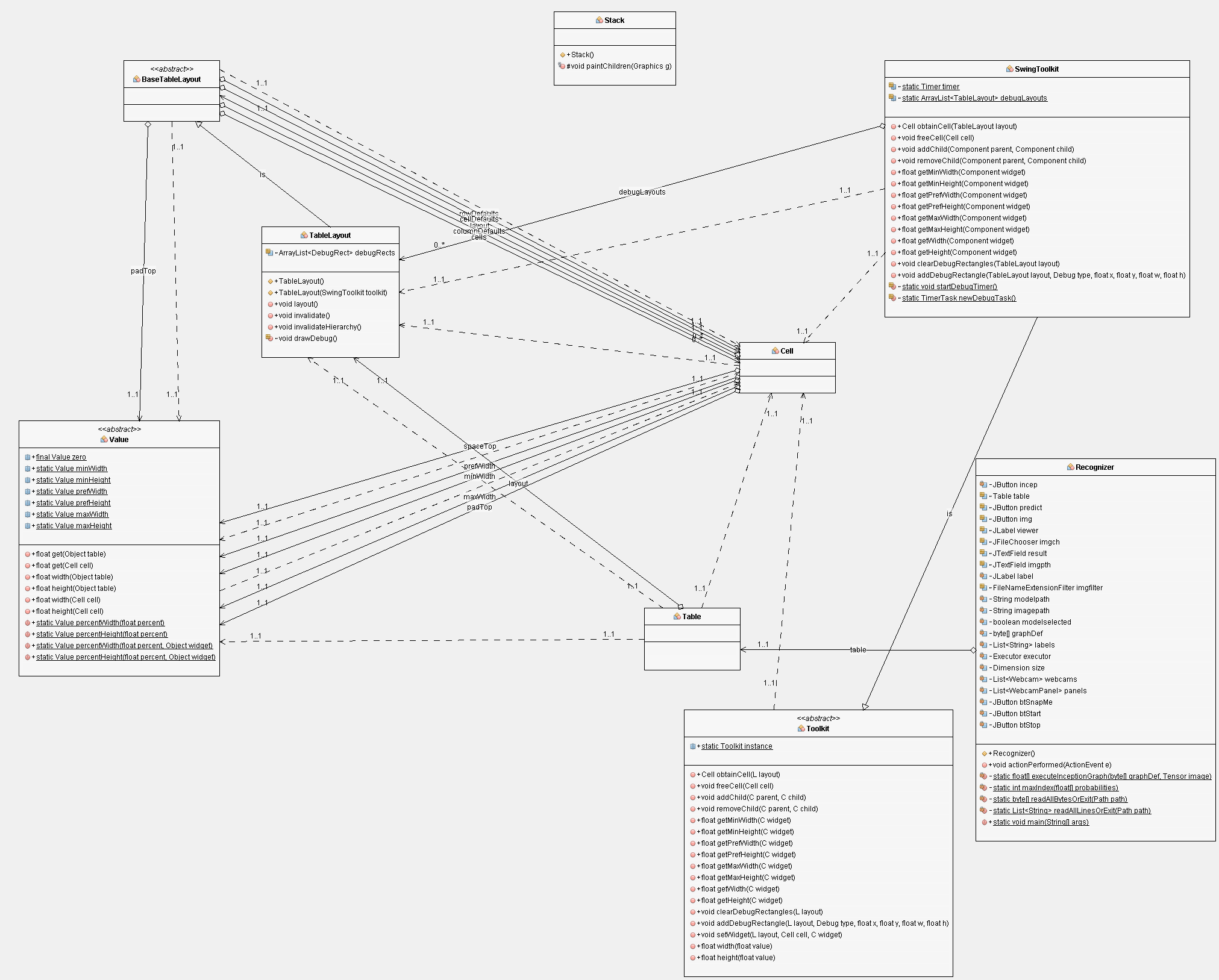
The system operates on the basis of two input methods which is importing random images and still camera images that are fed in to the system. Our model database consists of a variety of objects based on different categories. Methods used for object identification include 3D models, component identification, edge detection and analysis of appearances from different angles. When we feed in a random image to our object recognition system it will be able to identify it based on the likelihood of the object itself. If the likelihood percentage is 0 – 40% this indicates poor, 40 – 80% this indicates medium and 80 – 100% this indicates high likelihoods.

# **The UML Diagrams**

## Use Case Diagram

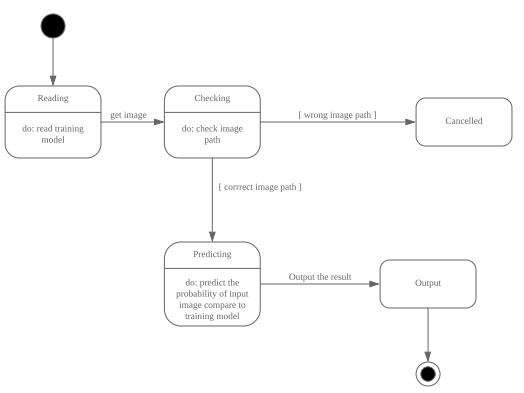


## Class Diagram



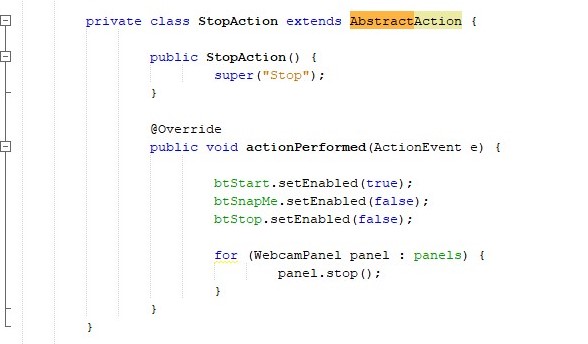
\*Complete Class Diagram refer to CD

## State Diagram



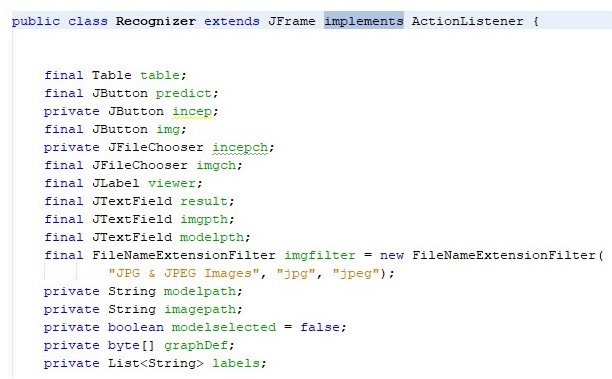
# **Detailed Descriptions**

Inheritance Method



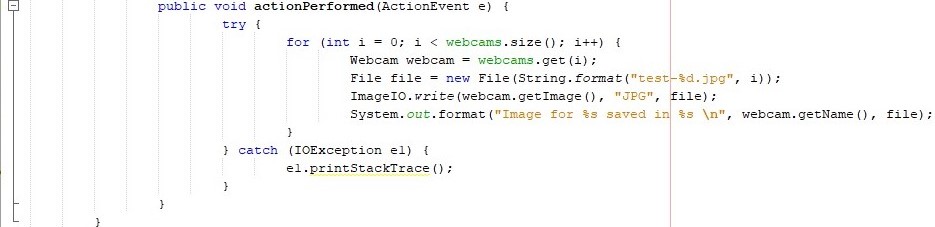
In OOP, we often organize classes in hierarchy to avoid duplication and reduce redundancy. The classes in the lower hierarchy inherit all the variables and methods from the higher hierarchies. A class in the upper hierarchy is called asuperclass. A class in the lower hierarchy is called a subclass. In the above figure the the StopAction class which is the subclass will inherit all the variables and method from the AbstractAction class which is the superclass.

Interface Method

****

An **interface in java** is a blueprint of a class. Interface specify what a class must do and not how. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship. An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

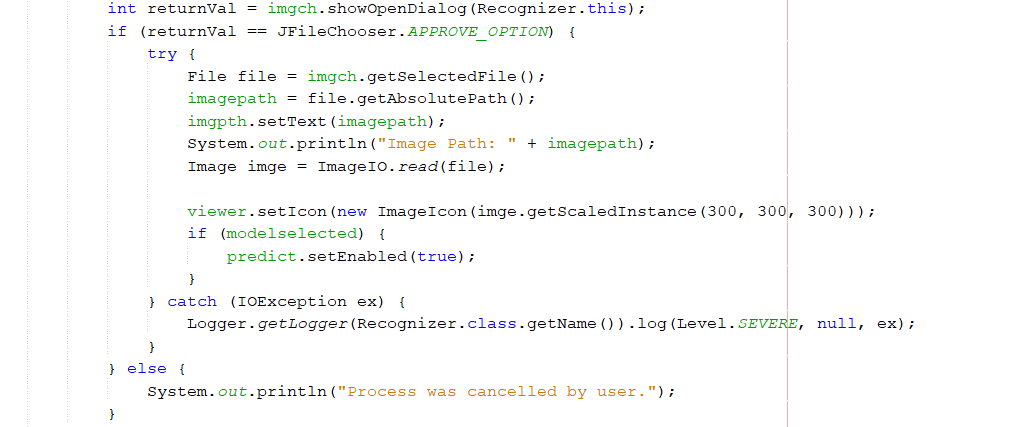
Exception Handling



When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things. When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception. The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block. The try and catch keywords come in pairs.

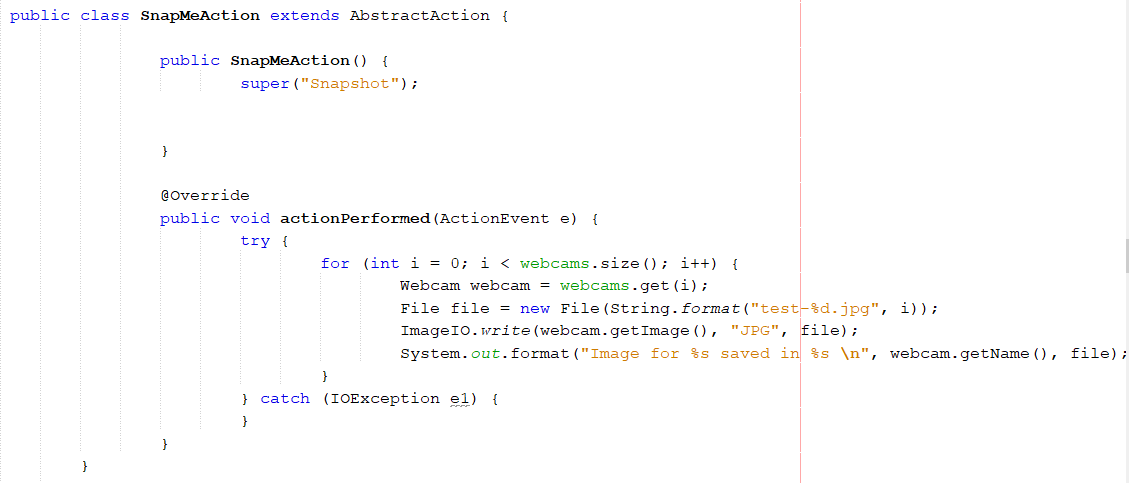
File Handling(Input/Output)

File input:



In our project, this is the example of file input. If the button is chosen, it will read an image file and print the path of the file. Then, print out the picture of the file.

File output:



In this example in our project, if we snap a picture, it will write/output a new file named test.jpg. Next, print out the name of the webcam name and the file output name.

# **Appendix**

## Source Code

Recognizer.java

package com.abc.ObjectRecognition;

import java.io.File;

import java.io.IOException;

import java.awt.Image;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.Files;

import java.nio.charset.Charset;

import java.util.List;

import java.util.Arrays;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.concurrent.Executor;

import java.util.concurrent.Executors;

import javax.imageio.ImageIO;

import javax.swing.JLabel;

import javax.swing.JFrame;

import javax.swing.JButton;

import javax.swing.ImageIcon;

import javax.swing.JTextField;

import javax.swing.JFileChooser;

import javax.swing.SwingUtilities;

import javax.swing.AbstractAction;

import javax.swing.filechooser.FileNameExtensionFilter;

import org.tensorflow.Graph;

import org.tensorflow.Output;

import org.tensorflow.Tensor;

import org.tensorflow.Session;

import org.tensorflow.DataType;

import com.github.sarxos.webcam.Webcam;

import com.github.sarxos.webcam.WebcamPanel;

import com.github.sarxos.webcam.WebcamResolution;

import com.esotericsoftware.tablelayout.swing.Table;

public class Recognizer extends JFrame implements ActionListener {

private final JButton incep= new JButton("Take A Picture");

final Table table =new Table();

final JButton predict = new JButton("Predict");

final JButton img= new JButton("Choose Image");

final JLabel viewer = new JLabel();

final JFileChooser imgch= new JFileChooser();

final JTextField result=new JTextField();

final JTextField imgpth;

private JLabel label;

final FileNameExtensionFilter imgfilter = new FileNameExtensionFilter(

"JPG & JPEG Images", "jpg", "jpeg");

private String modelpath;

private String imagepath;

private boolean modelselected = false;

private byte[] graphDef;

private List<String> labels;

public Recognizer() {

setTitle("Object Recognition System");

setSize(500, 500);

predict.setEnabled(false);

incep.addActionListener(this);

img.addActionListener(this);

predict.addActionListener(this);

imgch.setFileFilter(imgfilter);

imgch.setFileSelectionMode(JFileChooser.FILES\_ONLY);

imgpth = new JTextField();

imgpth.setEditable(false);

getContentPane().add(table);

table.addCell(incep).colspan(2);

table.row();

table.addCell(imgpth).width(250);

table.addCell(img);

table.row();

table.addCell(viewer).size(300, 300).colspan(2);

table.row();

table.addCell(predict).colspan(2);

table.row();

table.addCell(result).width(300).colspan(2);

table.row();

label = new JLabel();

label.setIcon(new ImageIcon("C:\\Users\\Asus\\Downloads\\Background.jpg"));

label.setBounds(0,0,500,500);

label.setHorizontalAlignment(JLabel.CENTER);

label.setVerticalAlignment(JLabel.CENTER);

table.setLayout(null);

table.add(label);

setLocationRelativeTo(null);

setResizable(false);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

File files = new File("C:\\Users\\Asus\\Downloads\\Training Model");

modelpath = files.getAbsolutePath();

System.out.println("Opening: " + files.getAbsolutePath());

modelselected = true;

graphDef = readAllBytesOrExit(Paths.get(modelpath, "tensorflow\_inception\_graph.pb"));

labels = readAllLinesOrExit(Paths.get(modelpath, "imagenet\_comp\_graph\_label\_strings.txt"));

if (e.getSource()==incep){

SnapMeAction snapMeAction= new SnapMeAction();

final JFrame window = new JFrame("Webcam");

for (Webcam webcam : webcams) {

webcam.setViewSize(size);

WebcamPanel panel = new WebcamPanel(webcam, size,false);

panel.setFPSDisplayed(true);

panel.setMirrored(true);

panels.add(panel);

add(panel);

window.add(panel);

btSnapMe.setEnabled(false);

btStop.setEnabled(false);

setLayout(new FlowLayout());

panel.add(btSnapMe);

panel.add(btStart);

panel.add(btStop);

window.setResizable(true);

window.pack();

window.setVisible(true);

}

}

else if (e.getSource() == img) {

int returnVal = imgch.showOpenDialog(Recognizer.this);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

try {

File file = imgch.getSelectedFile();

imagepath = file.getAbsolutePath();

imgpth.setText(imagepath);

System.out.println("Image Path: " + imagepath);

Image imge = ImageIO.read(file);

viewer.setIcon(new ImageIcon(imge.getScaledInstance(300, 300, 300)));

if (modelselected) {

predict.setEnabled(true);

}

} catch (IOException ex) {

Logger.getLogger(Recognizer.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

System.out.println("Process was cancelled by user.");

}

} else if (e.getSource() == predict) {

byte[] imageBytes = readAllBytesOrExit(Paths.get(imagepath));

try (Tensor image = Tensor.create(imageBytes)) {

float[] labelProbabilities = executeInceptionGraph(graphDef, image);

int bestLabelIdx = maxIndex(labelProbabilities);

result.setText("");

result.setText(String.format(

"BEST MATCH: %s (%.2f%% likely)",

labels.get(bestLabelIdx), labelProbabilities[bestLabelIdx] \* 100f));

System.out.println(

String.format(

"BEST MATCH: %s (%.2f%% likely)",

labels.get(bestLabelIdx), labelProbabilities[bestLabelIdx] \* 100f));

}

}

}

public class SnapMeAction extends AbstractAction {

public SnapMeAction() {

super("Snapshot");

}

@Override

public void actionPerformed(ActionEvent e) {

try {

for (int i = 0; i < webcams.size(); i++) {

Webcam webcam = webcams.get(i);

File file = new File(String.format("test-%d.jpg", i));

ImageIO.write(webcam.getImage(), "JPG", file);

System.out.format("Image for %s saved in %s \n", webcam.getName(), file);

}

} catch (IOException e1) {

}

}

}

private class StartAction extends AbstractAction implements Runnable {

public StartAction() {

super("Start");

}

@Override

public void actionPerformed(ActionEvent e) {

btStart.setEnabled(false);

btSnapMe.setEnabled(true);

executor.execute(this);

}

@Override

public void run() {

btStop.setEnabled(true);

for (WebcamPanel panel : panels) {

panel.start();

}

}

}

private class StopAction extends AbstractAction {

public StopAction() {

super("Stop");

}

@Override

public void actionPerformed(ActionEvent e) {

btStart.setEnabled(true);

btSnapMe.setEnabled(false);

btStop.setEnabled(false);

for (WebcamPanel panel : panels) {

panel.stop();

}

}

}

private final Executor executor = Executors.newSingleThreadExecutor();

private final Dimension size = WebcamResolution.VGA.getSize();

private final List<Webcam> webcams = Webcam.getWebcams();

private final List<WebcamPanel> panels = new ArrayList<>();

private final JButton btSnapMe = new JButton(new SnapMeAction());

private final JButton btStart = new JButton(new StartAction());

private final JButton btStop = new JButton(new StopAction());

private static float[] executeInceptionGraph(byte[] graphDef, Tensor image) {

try (Graph g = new Graph()) {

g.importGraphDef(graphDef);

try (Session s = new Session(g);

Tensor result = s.runner().feed("DecodeJpeg/contents", image).fetch("softmax").run().get(0)) {

final long[] rshape = result.shape();

if (result.numDimensions() != 2 || rshape[0] != 1) {

throw new RuntimeException(

String.format(

"Expected model to produce a [1 N] shaped tensor where N is the number of labels, instead it produced one with shape %s",

Arrays.toString(rshape)));

}

int nlabels = (int) rshape[1];

return result.copyTo(new float[1][nlabels])[0];

}

}

}

private static int maxIndex(float[] probabilities) {

int best = 0;

for (int i = 1; i < probabilities.length; ++i) {

if (probabilities[i] > probabilities[best]) {

best = i;

}

}

return best;

}

private static byte[] readAllBytesOrExit(Path path) {

try {

return Files.readAllBytes(path);

} catch (IOException e) {

System.err.println("Failed to read [" + path + "]: " + e.getMessage());

System.exit(1);

}

return null;

}

private static List<String> readAllLinesOrExit(Path path) {

try {

return Files.readAllLines(path, Charset.forName("UTF-8"));

} catch (IOException e) {

System.err.println("Failed to read [" + path + "]: " + e.getMessage());

System.exit(0);

}

return null;

}

static class GraphBuilder {

GraphBuilder(Graph g) {

this.g = g;

}

Output div(Output x, Output y) {

return binaryOp("Div", x, y);

}

Output sub(Output x, Output y) {

return binaryOp("Sub", x, y);

}

Output resizeBilinear(Output images, Output size) {

return binaryOp("ResizeBilinear", images, size);

}

Output expandDims(Output input, Output dim) {

return binaryOp("ExpandDims", input, dim);

}

Output cast(Output value, DataType dtype) {

return g.opBuilder("Cast", "Cast").addInput(value).setAttr("DstT", dtype).build().output(0);

}

Output decodeJpeg(Output contents, long channels) {

return g.opBuilder("DecodeJpeg", "DecodeJpeg")

.addInput(contents)

.setAttr("channels", channels)

.build()

.output(0);

}

Output constant(String name, Object value) {

try (Tensor t = Tensor.create(value)) {

return g.opBuilder("Const", name)

.setAttr("dtype", t.dataType())

.setAttr("value", t)

.build()

.output(0);

}

}

private Output binaryOp(String type, Output in1, Output in2) {

return g.opBuilder(type, type).addInput(in1).addInput(in2).build().output(0);

}

private final Graph g;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

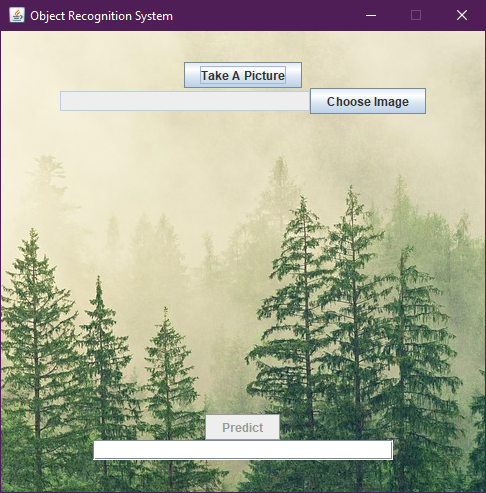
new Recognizer().setVisible(true);

});

}

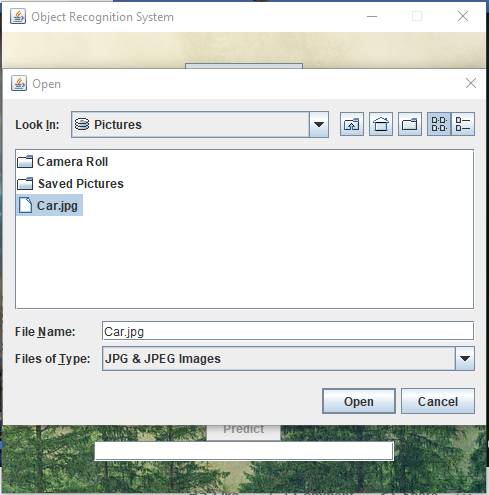
}

## Sample Interface Screen

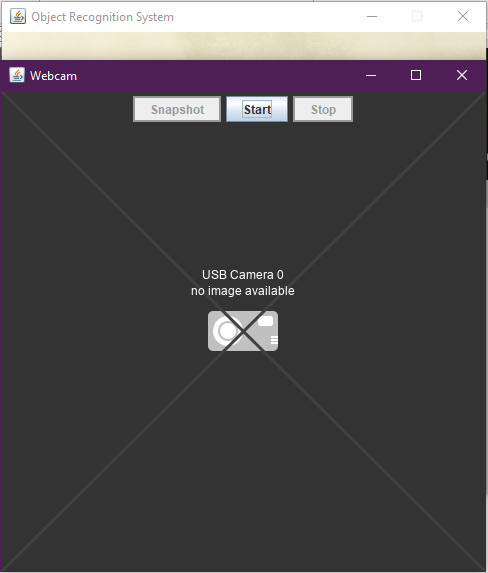


## Sample Input/Output Screens

Choose and select image



Take picture from webcam mode



Predict the probability of best match after selecting an image

